IN THE CLAIMS

At page 4 before the BRIEF DESCRIPTION OF THE DRAWINGS section, please insert the following:

BRIEF SUMMARY

A method and apparatus for efficient memory allocation and system management interrupt handling is herein described. In one embodiment, a single SMI is used to initialize each processor in a multi-processor environment. In addition, a location of a default SMI handler may be used as a wake-up vector to inactive processors to efficiently utilize memory. In another embodiment, unified handler code is executed on multiple processors to handle software generated SMIs.

At page 3, please make the following amendments to paragraph [0007]

[0007] However, these inefficient methods of initialization and handling are not limited to multiprocessor server systems, but exist in other systems, such as mobile multiprocessor systems. Hyper-Threading Technology (HT) is a technology from Intel.RTM. Corporation of Santa Clara, Calif. that enables execution of threads in parallel using a ~~signal~~ single physical processor. HT incorporates two logical processors on one physical processor (the same die). A logical processor is an independent processor visible to the operating system (OS), capable of executing code and maintaining a unique architectural state from other processor in a system. HT is achieved by having multiple architectural states that share one set of execution resources.

At page 7, please make the following amendments to paragraph [0023]

[0023] FIG. 1 illustrates a block diagram of a device 105 with multiple logical processors. A physical processor refers to a physical processor die or a single package. A logical processor is an

independent processor visible to the operating system (OS), capable of executing code and maintaining a unique architectural state from other processor in a system. Hyper-Threading Technology (HT) is a technology from Intel.RTM. Corporation of Santa Clara, Calif. that enables execution of threads in parallel using a ~~signal~~ single physical processor. HT includes two logical processors on one physical processor and is achieved by duplicating the architectural state, with each architecture state sharing one set of processor execution resources.

IN THE CLAIMS

Please amend claims 1-61 to the following:

1. (Currently Amended) A method comprising:

   receiving a first system management interrupt (SMI) with a first and a second processor;

   handling the first SMI with ~~a~~ the first processor;

   generating a wake-up signal with the first processor after receiving the first SMI, wherein the wake-up signal references a first memory address of a default SMI handler;

   receiving the wake-up signal with the second processor;

   awakening ~~a~~ the second processor, based on the wake-up signal from the first processor; and

   handling the first SMI with the second processor.

2. (Currently amended) The method of claim 1, wherein the first and second processors are logical processors, and wherein the first logical processor includes a first thread and the second logical processor includes a second thread.

3. (Original) The method of claim 1, wherein the first and second processors are physical processors.

4. (Previously Amended) The method of claim 1, wherein handling the first SMI with a first processor comprises: executing the default SMI handler located at the first memory address.

5. (Currently Amended) The method of claim 4, wherein the wake-up signal is a startup inter processor interrupt ~~SIPI~~ (SIPI) signal.

6. (Original) The method of claim 5, wherein the first memory address is aligned.

7. (Original) The method of claim 6, wherein the first memory address is 4k aligned.

8. (Currently Amended) The method of claim 5, wherein handling the first SMI with the second processor comprises executing the default SMI handler <u>with the second processor</u>.

9. (Original) The method of claim 8, wherein handling the first SMI with the second processor further comprises patching an instruction pointer to a second memory address.

10. (Original) The method of claim 9, wherein the second memory address is a non-aligned address.

11. (Previously Amended) The method of claim 10, further comprising: executing code at the second memory address after handling the first SMI with the second processor.

12. (Previously Amended) A method comprising:

receiving a first system management interrupt (SMI);

executing code at a first memory location with a first processor in response to the first SMI;

generating a wake-up signal with the first processor;

awakening a second processor, based on a wake-up signal from the first processor; and

executing the code from the first memory location with the second processor, in response to the first SMI after awakening the second processor.

13. (Previously Amended) The method of claim 12, wherein the wake-up signal is based on the first memory location.

14. (Previously Amended) The method of claim 13, wherein the code at the first memory location is default SMI handling code.

15. (Previously Amended) The method of claim 14, wherein the first memory location is located in conventional memory.

16. (Previously Amended) The method of claim 15, wherein the first memory location is aligned.

17. (Original) The method of claim 12, wherein both the first and second processors are logical processors located on the same die.

18. (Original) The method of claim 12, wherein the first and second processors are physical processors located on separate packages.

19. (Previously Amended) The method of claim 12, further comprising patching an instruction pointer for the second processor to a second memory location.

20. (Previously Amended) The method of claim 19, wherein the second memory location is a non-aligned location.

21. (Previously Amended) The method of claim 20, further comprising:
executing code at the second memory location after patching the instruction pointer to the second memory location.

22. (Previously Amended) The method of claim 12, further comprising generating the SMI before receiving the first SMI.

23. (Original) The method of claim 22, wherein generating the SMI comprises changing the logic level of a pin coupled to a controller hub.

24. (Original) The method of claim 22, wherein an APIC is used to generate the SMI.

25. (Original) The method of claim 24, wherein the APIC is located in the first processor.

26. (Currently Amended) A method comprising:

receiving a system management interrupt (SMI);

executing a SMI handler on a first processor to handle ~~a~~ the SMI for ~~a~~ the first processor; and

executing the SMI handler on a second processor to handle the SMI for ~~a~~ the second processor.

27. (Original) The method of claim 26, wherein the SMI is a software generated SMI.

28. (Original) The method of claim 27, wherein the first processor executes the SMI handler to handle the SMI for the first and the second processor.

29. (Original) The method of claim 26, wherein the SMI handler is located at a first memory address.

30. (Currently Amended) The method of claim 29, wherein the first memory address is a default offset from a first system management base (SMBase) address ~~for~~ associated with the first processor.

31. (Currently Amended) The method of claim 30, wherein executing the SMI handler to handle the SMI for the second processor comprises:

changing a target SMBase of the SMI handler from the first SMBase address to a second SMBase address ~~for~~ associated with the second processor; and

executing the ~~first~~ SMI handler using the second SMBase as the target SMBase.

32. (Original) A method comprising:

executing system management interrupt (SMI) code with a first processor to handle a SMI for the first processor;

checking if the SMI is a software generated SMI; and

executing the SMI code to handle the SMI for a second processor, if the SMI is software generated.

33. (Original) The method of claim 32, wherein the first processor executes the SMI code to handle the SMI for the second processor, if the SMI is software generated.

34. (Original) The method of claim 32, wherein the second processor executes the SMI code to handle the SMI for the second processor, if the SMI is software generated.

35. (Original) The method of claim 32, wherein the first processor has a first system management base (SMBase) address.

36. (Previously Amended) The method of claim 35, wherein the second processor has a second SMBase address.

37. (Original) The method of claim 36, wherein said SMI code is located at first memory location, which has an offset from the first SMBase address.

38. (Previously Amended) The method of claim 37, wherein executing said SMI code to handle the SMI for the second processor comprises:

changing a target SMBase from the first SMBase to the second SMBase; and

executing the SMI code using the second SMBase as the target SMBase.

39. (Original) The method of claim 38, further comprising returning the target SMBase of the SMI handler to the first SMBase after executing the SMI code to handle the SMI for the second processor.

40. (Previously Amended) An apparatus comprising:

a controller to generate a first system management interrupt (SMI);

a first logical processor, coupled to the controller, to handle the first SMI and generate a wake-up signal, wherein the wake-up signal references a first memory address of a default SMI handler; and

a second logical processor, coupled to the controller, to handle the first SMI after the wake-up signal is received from the first logical processor.

41. (Previously Amended) The apparatus of claim 40, wherein handling the first SMI with the first logical processor comprises executing the default SMI handler with the first logical processor.

42. (Previously Amended) The apparatus of claim 41, wherein the first memory address is 1k aligned.

43. (Previously Amended) The apparatus of claim 41, wherein handling the first SMI with the second logical processor comprises executing the default SMI handler with the second logical processor.

44. (Original) The apparatus of claim 43, wherein handling the first SMI with the second logical processor further comprises patching an instruction pointer to a second memory address.

45. (Original) A system comprising:

a controller hub to generate a first system management interrupt (SMI);

a memory with a first memory address that contains code;

a first processor coupled to the controller hub to handle the first SMI, wherein the first processor executes the code at the first memory address and generates a wake-up signal; and

a second processor coupled to the controller hub to handle the first SMI after receiving the wake-up signal, wherein the second processor executes the code at the first memory address.

46. (Original) The system of claim 45, wherein the first and second processors are logical processors on the same die.

47. (Original) The system of claim 45, wherein the first and second processors are physical processors located on separate packages.

48. (Original) The system of claim 45, wherein a pin is toggled on the controller hub to generated the first SMI.

49. (Original) The system of claim 45, wherein code is executed by the controller hub to generate the first SMI.

50. (Original) The system of claim 45, wherein the code at the first memory address is SMI handling code.

51. (Original) The system of claim 50, wherein the wake-up signal is a vector containing the first memory address.

52. (Original) The system of claim 51, wherein handling the first SMI with the second processor after receiving the wake-up signal further comprises setting a pointer to a second memory address.

53. (Original) The system of claim 52, wherein the second processor, upon resuming from handling the SMI, executes code at the second memory address.

54. (Previously Amended) A system comprising:

a memory with a first memory address having system management interrupt (SMI) code;

a first processor to execute the SMI code when a SMI is received; and

a second processor to execute the SMI code, if the SMI is software generated.

55. (Original) The system of claim 54, wherein the first processor has a first system management base (SMBase) address.

56. (Original) The system of claim 55, wherein the second processor has a second SMBase address.

57. (Original) The system of claim 56, wherein the first memory address has an offset from the first SMBase.

58. (Previously Amended) The system of claim 57, wherein a target SMBase referenced by the SMI code, by default, is the first SMBase.

59. (Original) The system of claim 58, wherein the target SMBase is changed to the second SMBase before the second processor executes the SMI code.

60. (Original) The system of claim 54, wherein the first and second processors are logical processors.

61. (Original) The system of claim 54, wherein the first and second processors are physical processors.